Linux内核锁正确使用和调试

讲解时间：2月3日晚9点
宋宝华 <21cnbao@gmail.com>

微信群直播：
https://mp.weixin.qq.com/s/yyN_JkJ2bo9K-f0vTGLI4g

扫描二维码报名

Linux任督二脉

（学习形式：微信群）

麦当劳喜欢您来，喜欢您再来

扫描关注

Linuxer

大纲

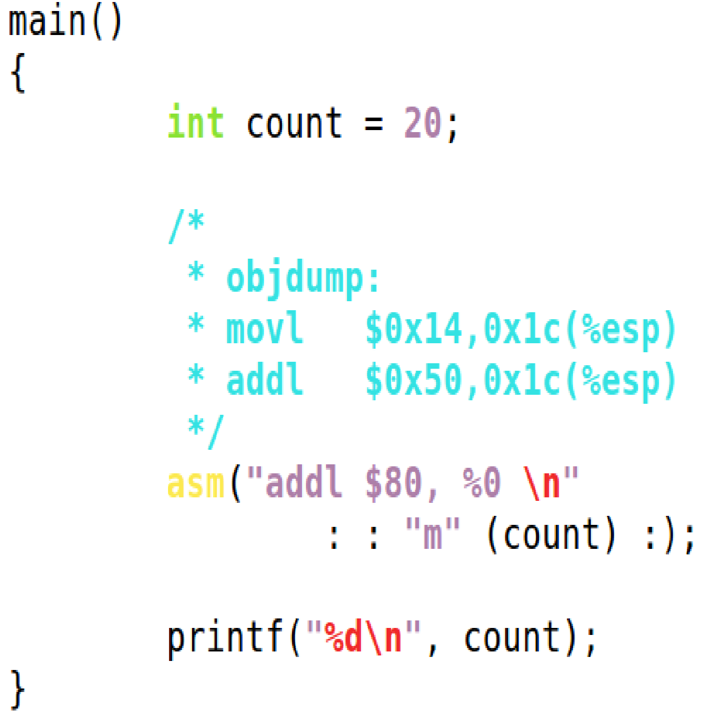
*原子性
*RMW
*atomic
*锁住语义整体
*spinlock
*irq关
*mutex
*lockup detector

+1也不是原子的(RISC vs. CISC)

- 哪怕一个整数的+1，也不是原子的。要经过读-修改-写

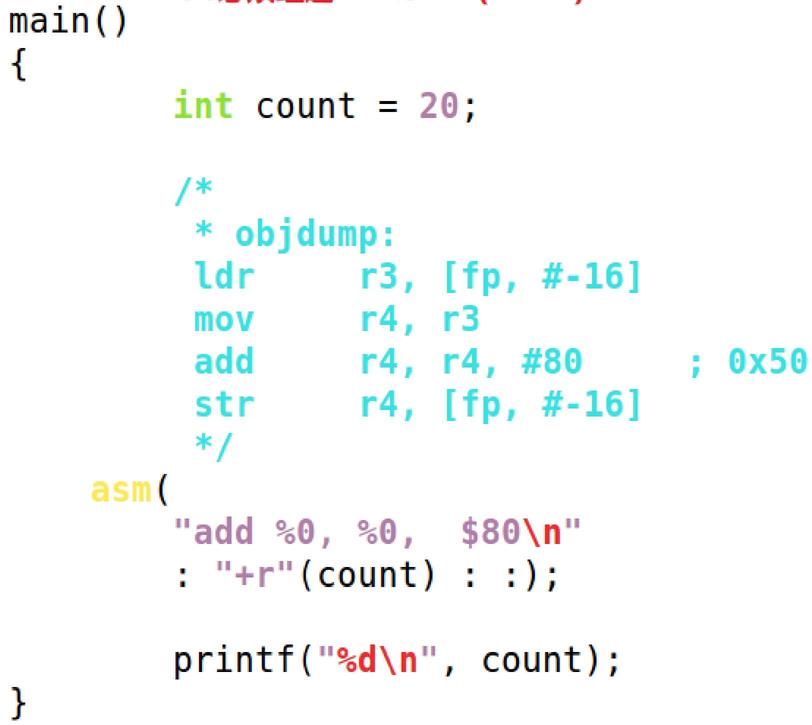
RMW !!!

CSIC处理器，可以直接在内存上面做加法



```
main()
{
        int count = 20;

        /*
         * objdump:
         * movl   $0x14,0x1c(%esp)
         * addl   $0x50,0x1c(%esp)
         */
        asm("addl $80, %0 \n"
                : : "m" (count) :);

        printf("%d\n", count);
}
```

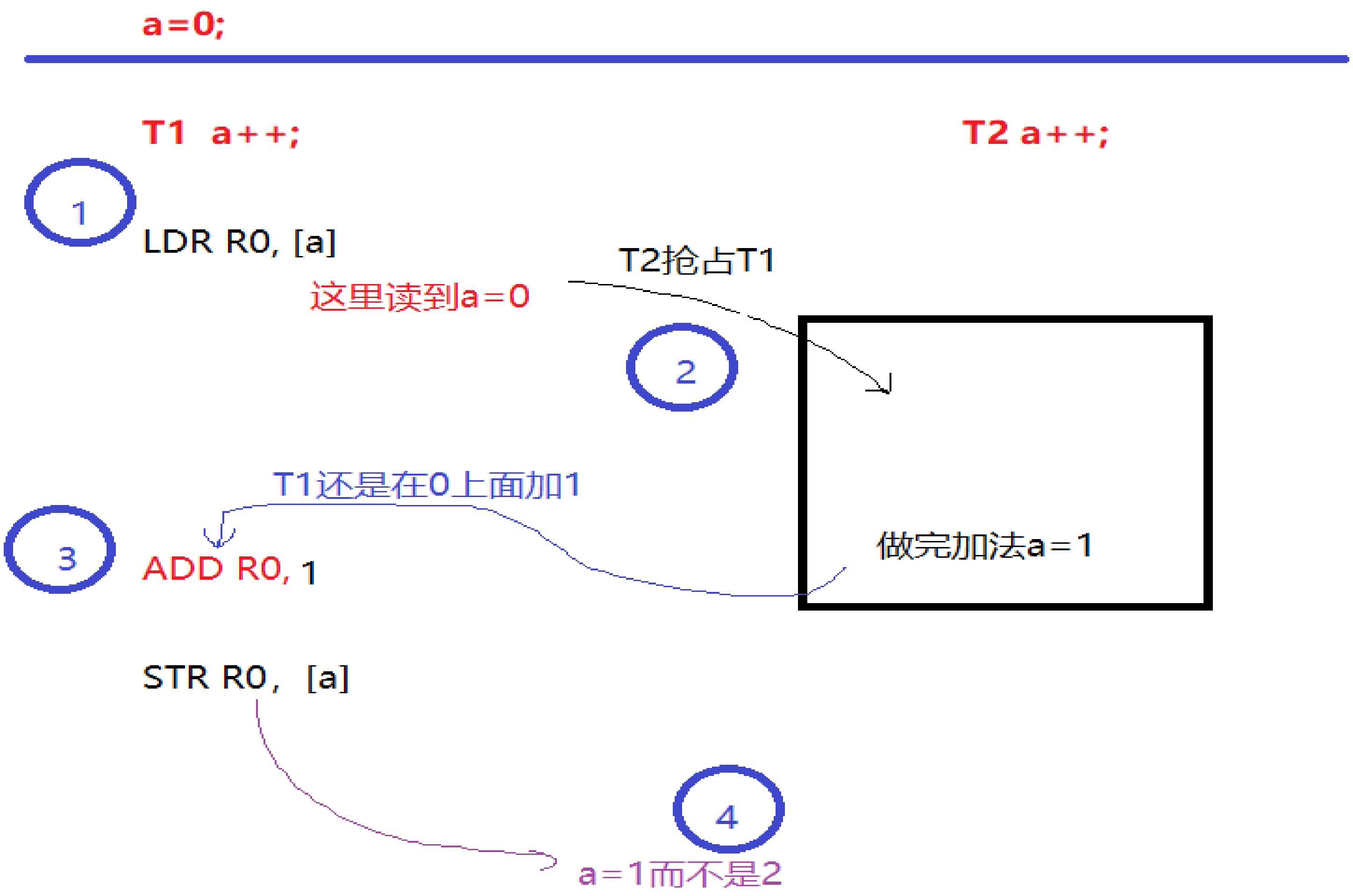
RISC必须经过LDR.STR（RMW）



```
main()
{
        int count = 20;

        /*
         * objdump:
         ldr      r3, [fp, #-16]
         mov      r4, r3
         add      r4, r4, #80       ; 0x50
         str      r4, [fp, #-16]
         */
    asm(
        "add %0, %0,  $80\n"
        : "+r"(count) : :);

        printf("%d\n", count);
}
```

读-修改-写可能引发的问题

- 两个线程同时对变量a做加法



a=0;

T1 a++;

T2 a++;

LDR R0, [a]

这里读到a=0

T2抢占T1

T1还是在0上面加1

ADD R0, 1

做完加法a=1

STR R0, [a]

a=1而不是2

atomic

- atomic保证整数操作的原子性

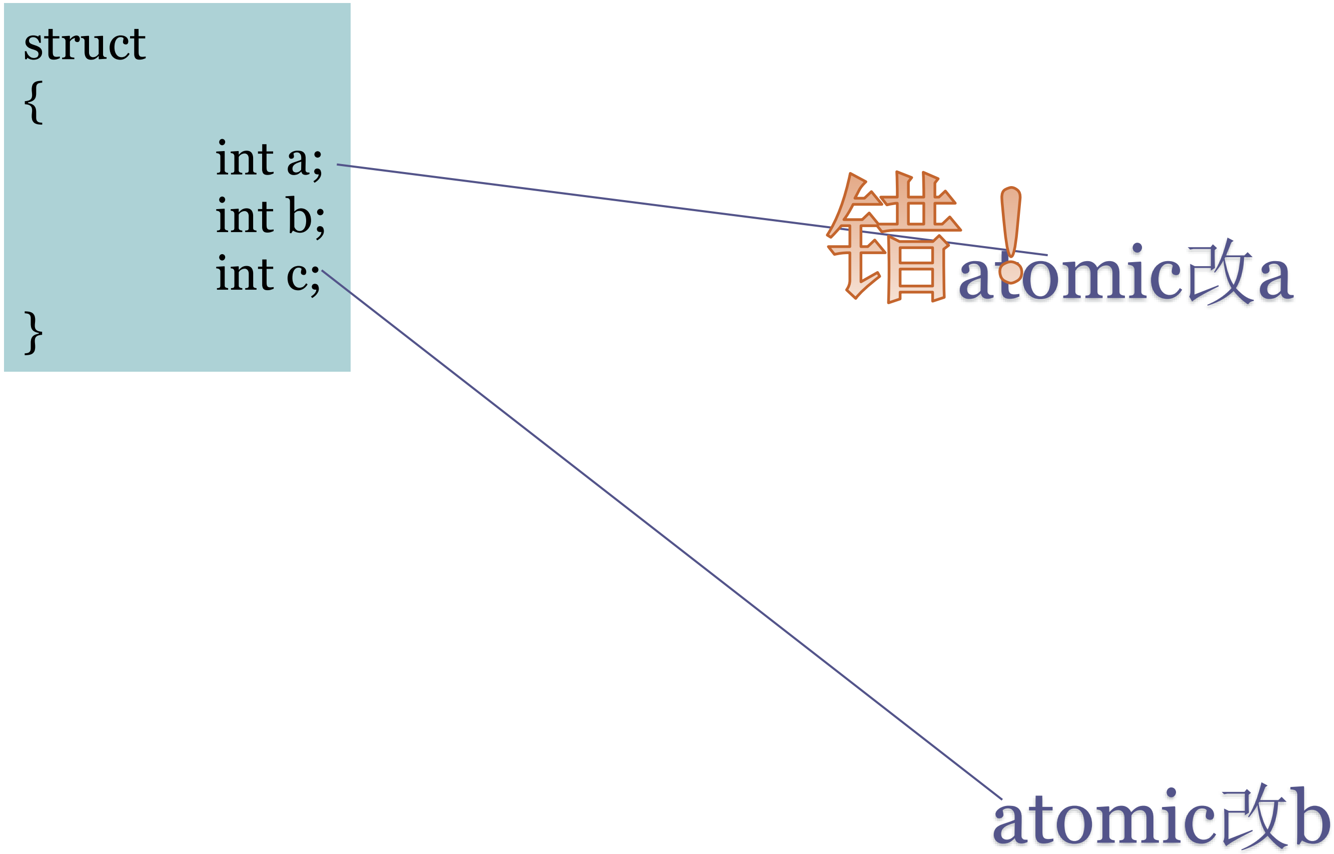
```
void atomic_add(int i, atomic_t *v)
void atomic_sub(int i, atomic_t *v)
void atomic_inc(atomic_t *v)
void atomic_dec(atomic_t *v)
....
```

两个RMW序列只有1个能成功

```
 MOV r1, #0x1                ; load the ‘lock taken’ value
try:
   LDREX r0, [LockAddr]       ; load the lock value
   CMP r0, #0                 ; is the lock free?
   STREXEQ r0, r1, [LockAddr]  ; try and claim the lock
   CMPEQ r0, #0               ; did this succeed?
   BNE try                    ; no – try again
   ....                       ; yes – we have the lock
```

atomic不能这么用

- 看到一个结构体里面都是int，就用atomic



```
struct
{
        int a;
        int b;
        int c;
}
```

错！

atomic改a

atomic改b

设想2个学生

- 张三 男
- 李四 女

改姓名和性别都加锁
一定不会出现

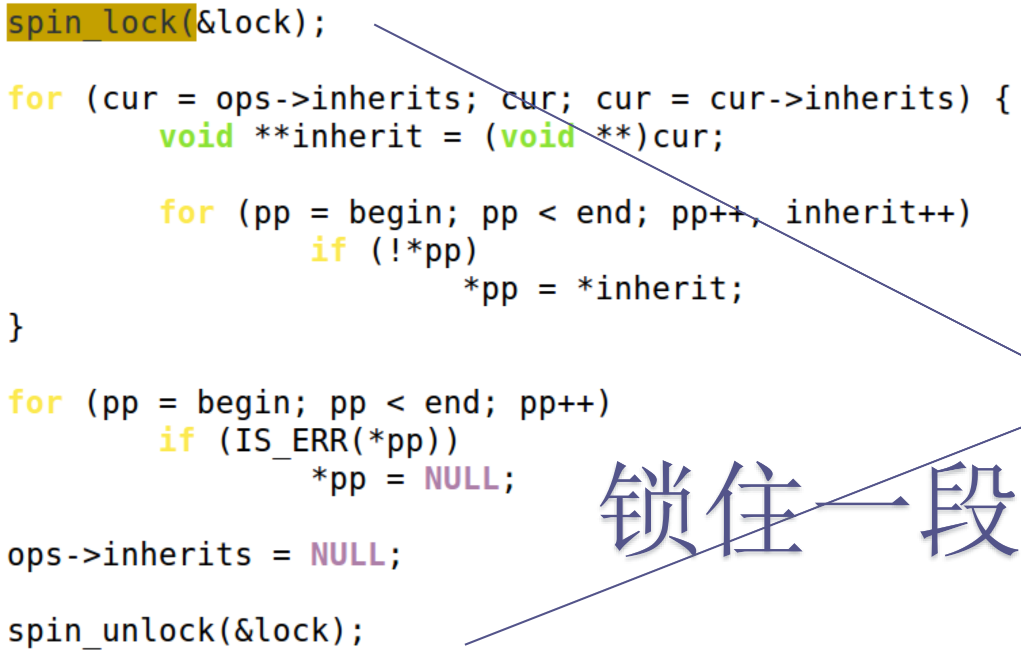
- 张四
- 李三

但是，可能出现

- 张三 女
- 李四 男

一定要锁住一个语义完整的整体

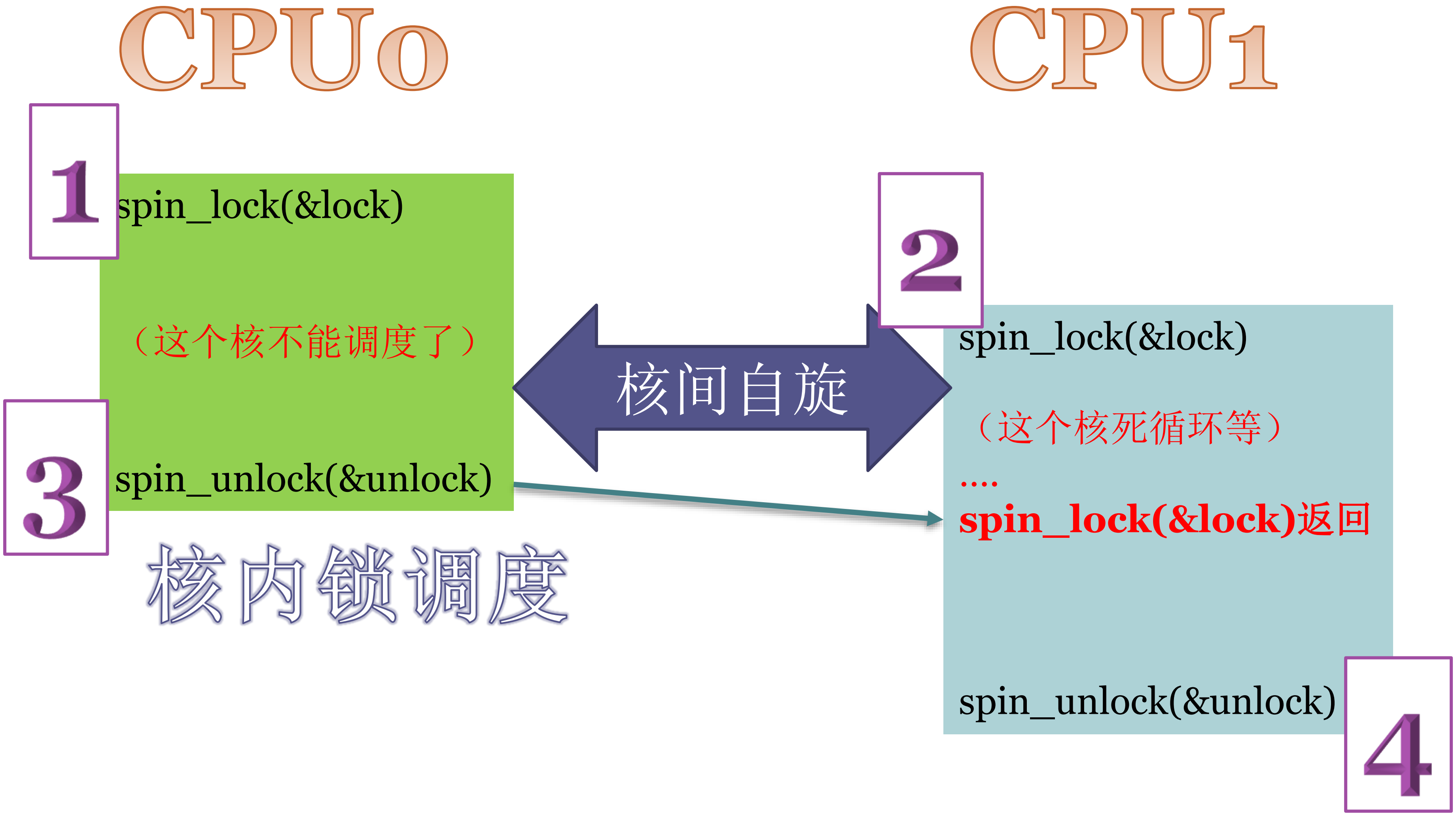
spinlock



```
spin_lock(&lock);

for (cur = ops->inherits; cur; cur = cur->inherits) {
        void **inherit = (void **)cur;

        for (pp = begin; pp < end; pp++, inherit++)
                if (!*pp)
                        *pp = *inherit;
}

for (pp = begin; pp < end; pp++)
        if (IS_ERR(*pp))
                *pp = NULL;

ops->inherits = NULL;

spin_unlock(&lock);
```

锁住一段不能睡眠的区间

spinlock干了什么



CPU0

CPU1

1

spin_lock(&lock)

（这个核不能调度了）

3

spin_unlock(&unlock)

核内锁调度

核间自旋

2

spin_lock(&lock)

（这个核死循环等）

….

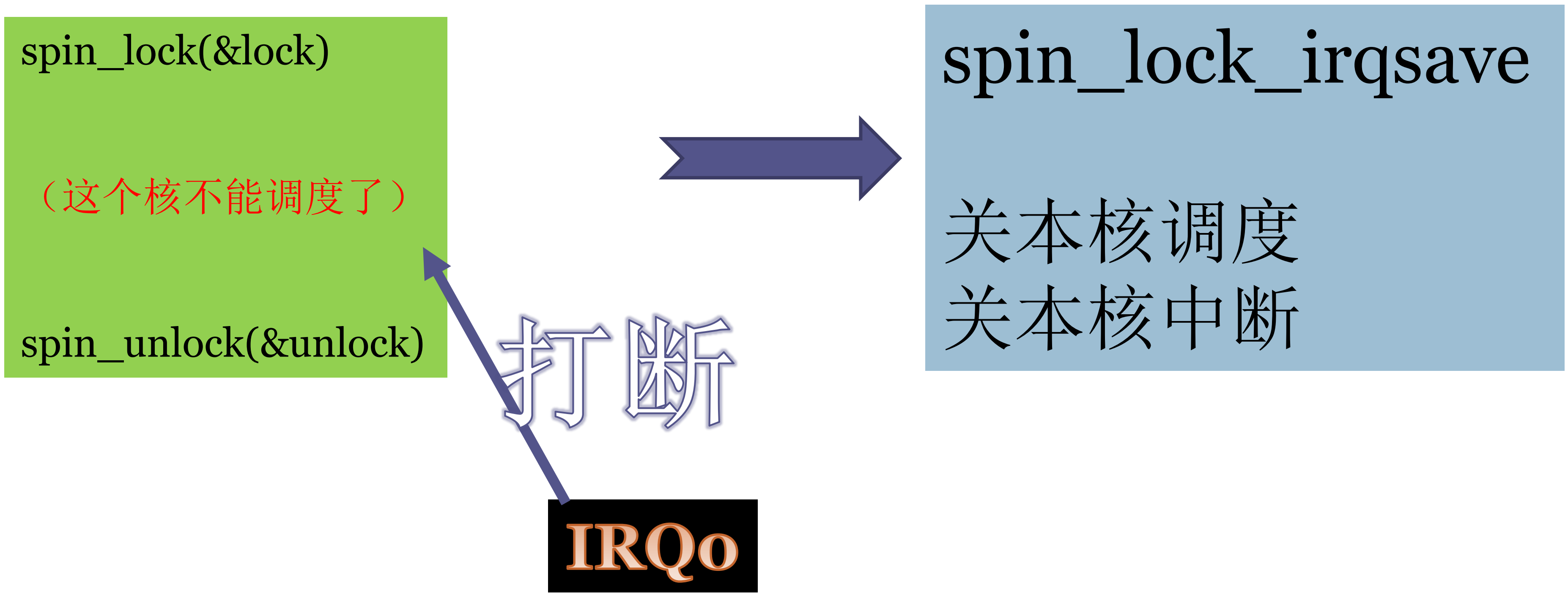
spin_lock(&lock)返回

spin_unlock(&unlock)

4

来了中断怎么办?

CPUo



spin_lock(&lock)

（这个核不能调度了）

spin_unlock(&unlock)

打断

IRQo

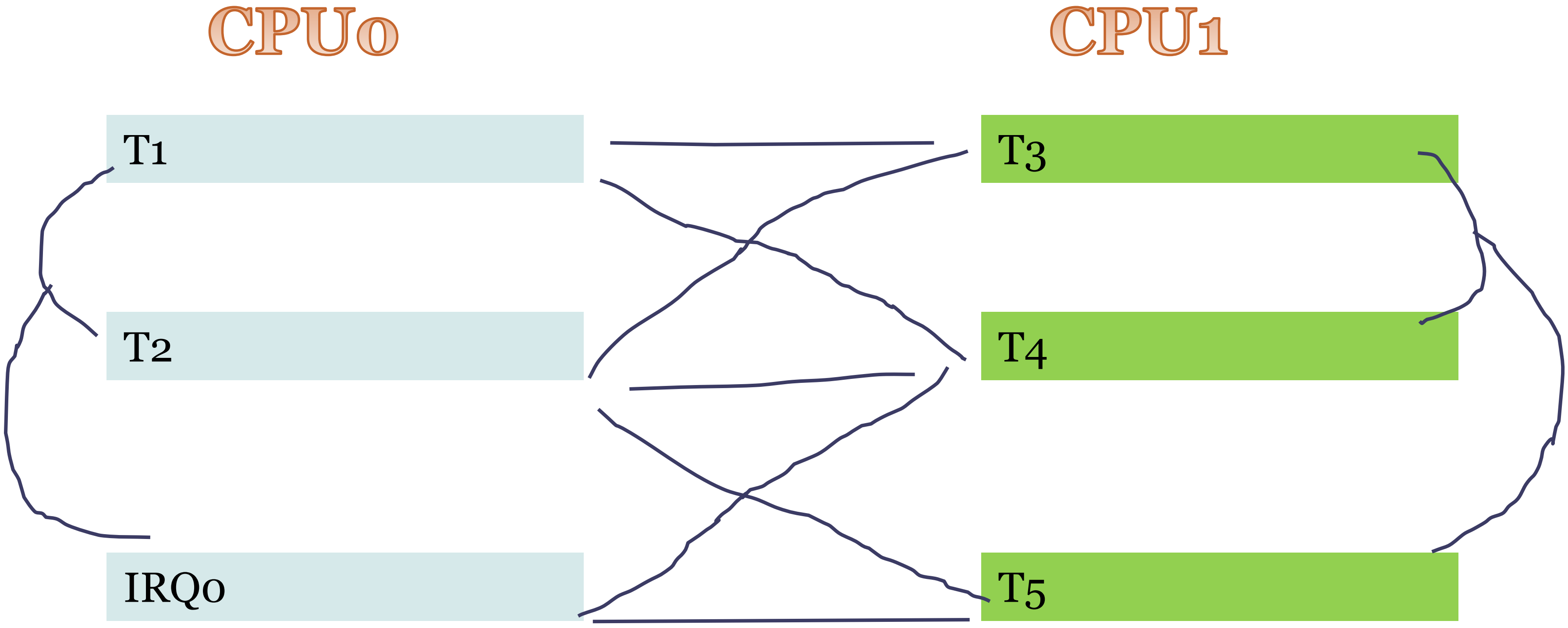
spin_lock_irqsave

关本核调度

关本核中断

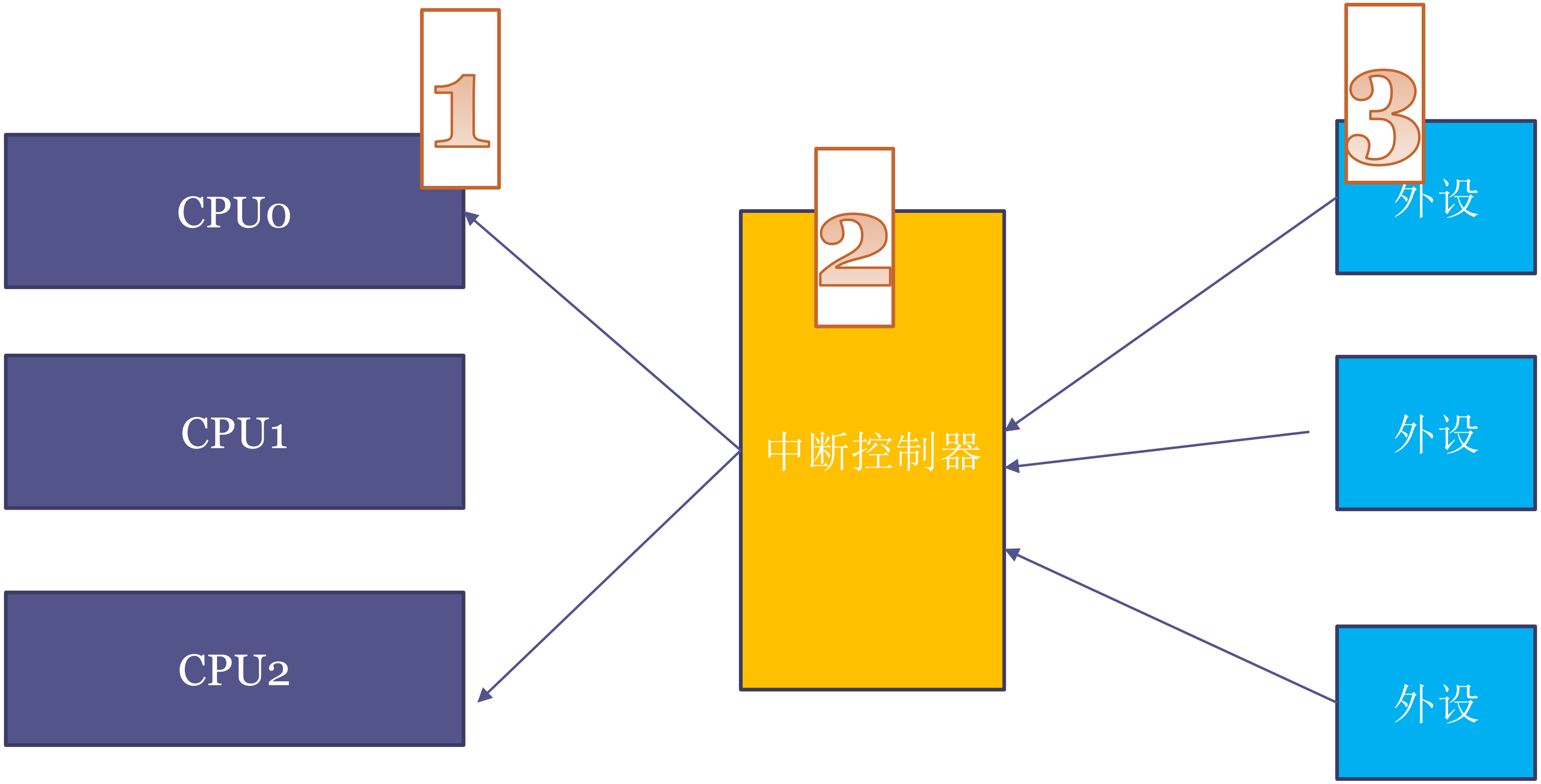
到处都有risk

线程与线程
线程与中断
中断与中断
此核与彼核



local_irq_disable/save和irq_disable

local_irq_disable/save：让本CPU不响应所有中断(掐断位置1)
irq_disable：让某中断不能发给所有CPU(掐断位置2)



消除全部的risk

	CPU0	CPU1	
T0	spin_lock_irqsave … spin_unlock_irqrestore	spin_lock_irqsave … spin_unlock_irqrestore	T3
T1	spin_lock_irqsave … spin_unlock_irqrestore	spin_lock_irqsave … spin_unlock_irqrestore	T4
IRQ0	spin_lock … spin_unlock	spin_lock … spin_unlock	IRQ1

区别表

	核内	核间
spin_lock	锁住调度	自旋
local_irq_disable/save	锁住中断 锁住调度	没有意义
spin_lock_irqsave	锁住中断 锁住调度	自旋

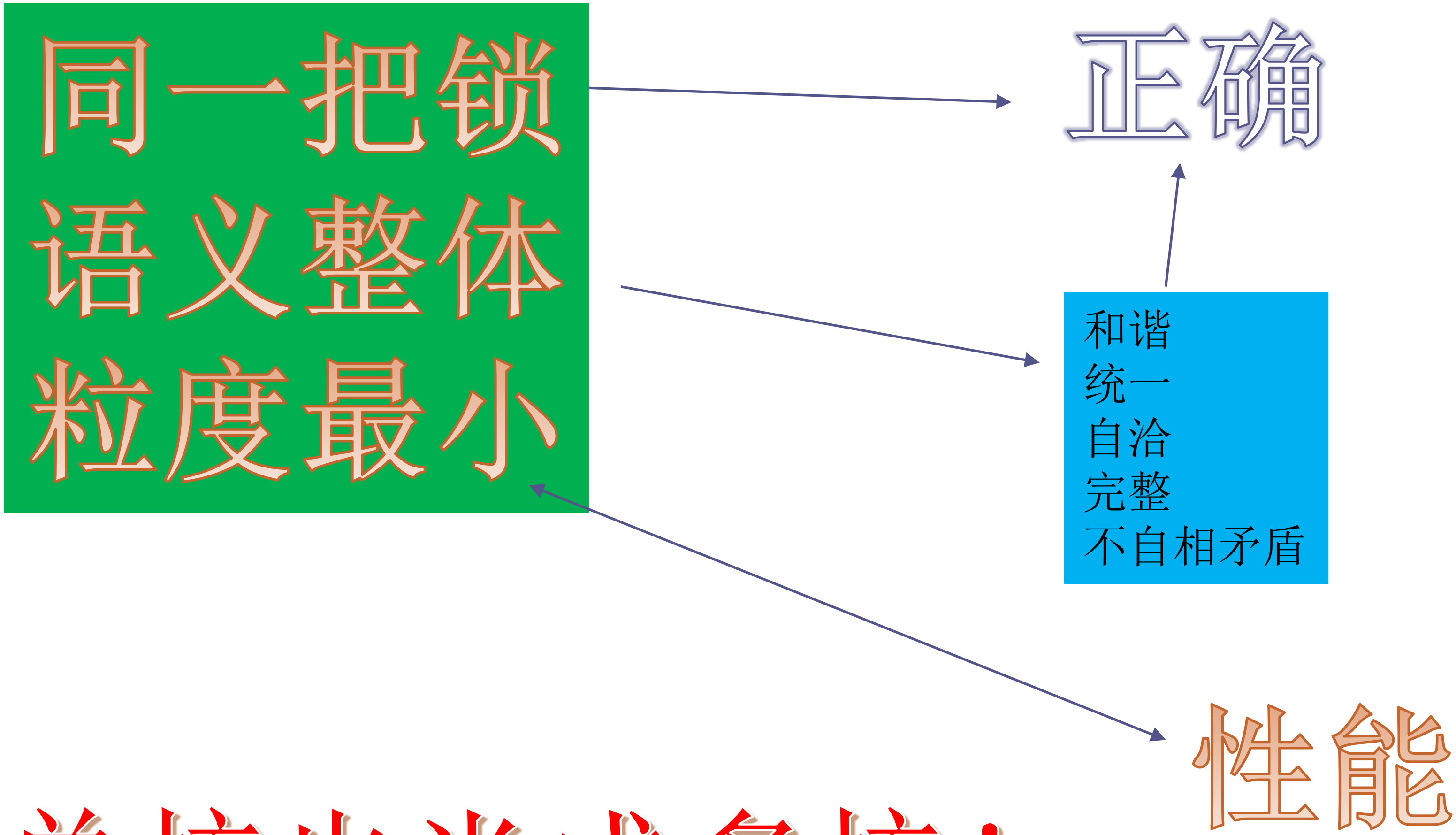
调用local_irq_disable 多半是个bug

mutex

- T1拿到了mutex，如果T2拿不到，T2睡眠；
- T1释放mutex,T2被唤醒

```
mutex_lock(&lock);


....
(中间可以睡眠)



mutex_unlock(&lock);
```

加锁原则



同一把锁
语义整体
粒度最小

正确

和谐
统一
自洽
完整
不自相矛盾

性能

单核也当成多核！

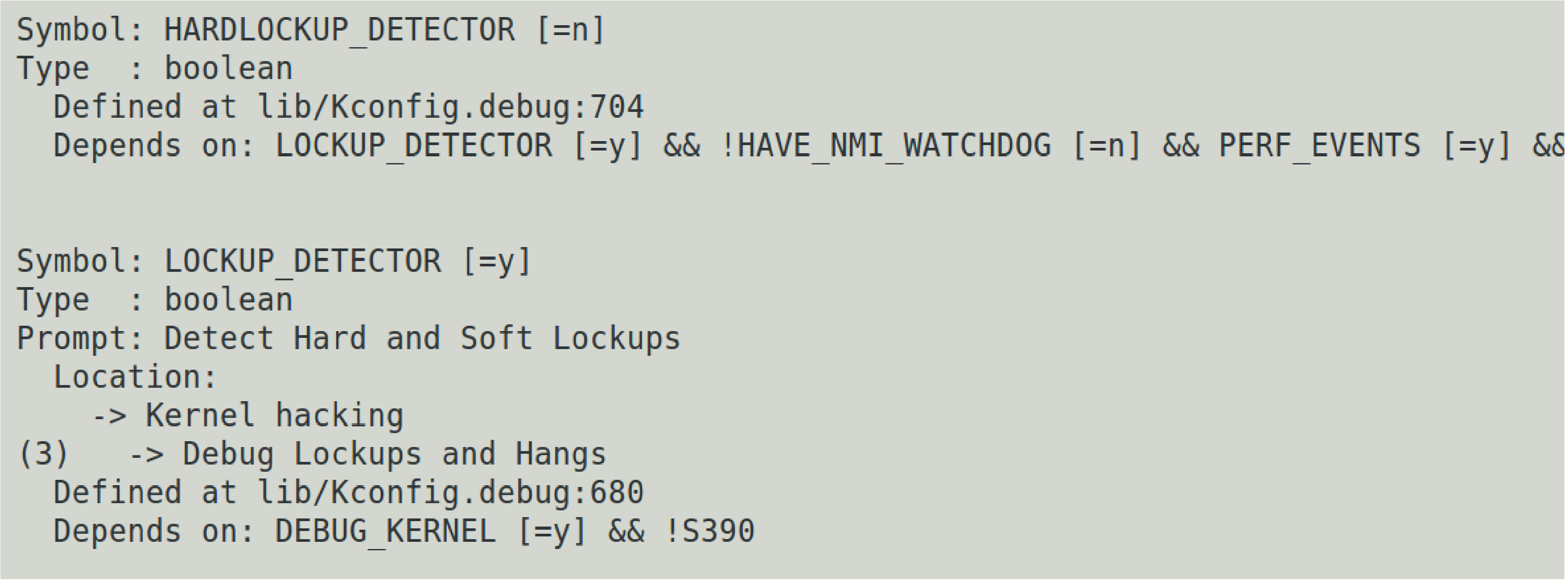
Lockup detector

- kernel/watchdog.c

✓ NMI中断 + 定时器中断+高优先级RT线程

✓ 用定时器中断，检测高优先级线程有无机会执行->soft lockup

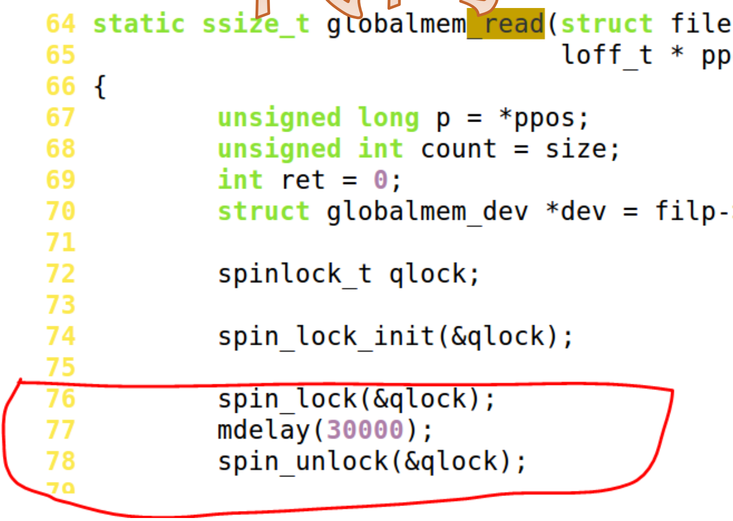
✓ 用NMI，检测定时器中断有无机会执行 -> hard lockup



```
Symbol: HARDLOCKUP_DETECTOR [=n]
Type  : boolean
  Defined at lib/Kconfig.debug:704
  Depends on: LOCKUP_DETECTOR [=y] && !HAVE_NMI_WATCHDOG [=n] && PERF_EVENTS [=y] &&


Symbol: LOCKUP_DETECTOR [=y]
Type  : boolean
Prompt: Detect Hard and Soft Lockups
  Location:
    -> Kernel hacking
(3)   -> Debug Lockups and Hangs
  Defined at lib/Kconfig.debug:680
  Depends on: DEBUG_KERNEL [=y] && !S390
```

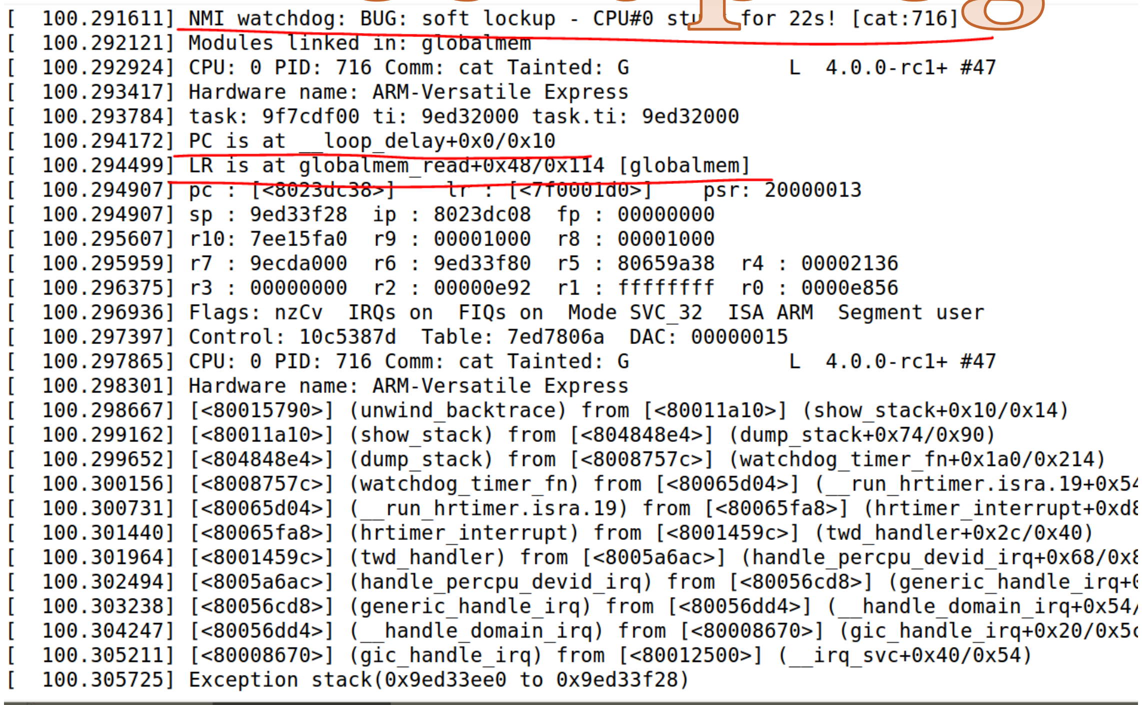
Lockup detector案 例

代码



```
static ssize_t globalmem_read(struct file *filp, char
                              loff_t * ppos)
{
        unsigned long p = *ppos;
        unsigned int count = size;
        int ret = 0;
        struct globalmem_dev *dev = filp->private_dat

        spinlock_t qlock;

        spin_lock_init(&qlock);

        spin_lock(&qlock);
        mdelay(30000);
        spin_unlock(&qlock);
```

Lockup log



```
[  100.291611] NMI watchdog: BUG: soft lockup - CPU#0 stuck for 22s! [cat:716]
[  100.292121] Modules linked in: globalmem
[  100.292924] CPU: 0 PID: 716 Comm: cat Tainted: G             L  4.0.0-rc1+ #47
[  100.293417] Hardware name: ARM-Versatile Express
[  100.293784] task: 9f7cdf00 ti: 9ed32000 task.ti: 9ed32000
[  100.294172] PC is at __loop_delay+0x0/0x10
[  100.294499] LR is at globalmem_read+0x48/0x114 [globalmem]
[  100.294907] pc : [<8023dc38>]    lr : [<7f0001d0>]    psr: 20000013
[  100.294907] sp : 9ed33f28  ip : 8023dc08  fp : 00000000
[  100.295607] r10: 7ee15fa0  r9 : 00001000  r8 : 00001000
[  100.295959] r7 : 9ecda000  r6 : 9ed33f80  r5 : 80659a38  r4 : 00002136
[  100.296375] r3 : 00000000  r2 : 00000e92  r1 : ffffffff  r0 : 0000e856
[  100.296936] Flags: nzCv  IRQs on  FIQs on  Mode SVC_32  ISA ARM  Segment user
[  100.297397] Control: 10c5387d  Table: 7ed7806a  DAC: 00000015
[  100.297865] CPU: 0 PID: 716 Comm: cat Tainted: G             L  4.0.0-rc1+ #47
[  100.298301] Hardware name: ARM-Versatile Express
[  100.298667] [<80015790>] (unwind_backtrace) from [<80011a10>] (show_stack+0x10/0x14)
[  100.299162] [<80011a10>] (show_stack) from [<804848e4>] (dump_stack+0x74/0x90)
[  100.299652] [<804848e4>] (dump_stack) from [<8008757c>] (watchdog_timer_fn+0x1a0/0x214)
[  100.300156] [<8008757c>] (watchdog_timer_fn) from [<80065d04>] (__run_hrtimer.isra.19+0x54
[  100.300731] [<80065d04>] (__run_hrtimer.isra.19) from [<80065fa8>] (hrtimer_interrupt+0xd8
[  100.301440] [<80065fa8>] (hrtimer_interrupt) from [<8001459c>] (twd_handler+0x2c/0x40)
[  100.301964] [<8001459c>] (twd_handler) from [<8005a6ac>] (handle_percpu_devid_irq+0x68/0x8
[  100.302494] [<8005a6ac>] (handle_percpu_devid_irq) from [<80056cd8>] (generic_handle_irq+0
[  100.303238] [<80056cd8>] (generic_handle_irq) from [<80056dd4>] (__handle_domain_irq+0x54/
[  100.304247] [<80056dd4>] (__handle_domain_irq) from [<80008670>] (gic_handle_irq+0x20/0x5c
[  100.305211] [<80008670>] (gic_handle_irq) from [<80012500>] (__irq_svc+0x40/0x54)
[  100.305725] Exception stack(0x9ed33ee0 to 0x9ed33f28)
```

更早课程

- 《Linux总线、设备、驱动模型》录播：http://edu.csdn.net/course/detail/5329

- 深入探究Linux的设备树 http://edu.csdn.net/course/detail/5627

- Linux进程、线程和调度 http://edu.csdn.net/course/detail/5995

- C语言大型软件设计的面向对象 https://edu.csdn.net/course/detail/6496

谢 谢！